

Breviar

Laborator 10 - Interfete grafice

- Responsabil: Mihai Nan [mailto:mihai.nan.cti@gmail.com]
- Profesor titular: Carmen Odubășteanu

Introducere

În cazul programelor pe care le-am făcut până acum, toate mesajele și răspunsurile apăreau ca linii de text sugestive, ecranul fiind folosit în mod text. Un astfel de stil de comunicare nu este atractiv pentru utilizatori, motiv pentru care se prefera dialogul prin interfețe grafice sau **GUI (Graphical User Interface)**, ecranul fiind folosit în mod grafic.

În trecerea de la o versiune la alta, bibliotecile de clase care oferă servicii grafice au suferit, probabil, cele mai mari schimbări. Acest lucru se datorează, pe de o parte, dificultăți legate de implementarea noțiunii de portabilitate, iar pe de altă parte nevoii de a integra mecanismele GUI cu tehnologii aparute și dezvoltate ulterior.

În momentul actual, există două modalități de a crea o aplicație cu interfața grafică, iar acestea sunt:

- **AWT (Abstract Windowing Toolkit)** - este API-ul inițial pus la dispoziție începând cu primele versiuni de Java;
- **Swing** - este parte dintr-un proiect mai amplu numit **JFC (Java Foundation Classes)** creat în urma colaborării dintre Sun, Netscape și IBM, care se bazează pe modelul **AWT**, extinzând funcționalitatea acestuia și adăugând sau înlocuind unele componente pentru dezvoltarea aplicațiilor GUI.

Este preferabil ca aplicațiile Java să fie create folosind tehnologia Swing, deoarece aceasta pune la dispoziție o paletă mult mai largă de facilități, însă nu se va renunța complet la AWT, deoarece aici există clase esențiale, reutilizate în Swing.

Pachetul Swing

Componentele **Swing**, spre deosebire de predecesoarele din versiunile Java anterioare, sunt implementate în întregime în Java. Aceasta are ca rezultat o mai bună compatibilitate cu platforme diferite decât în cazul folosirii componentelor **AWT**. Unul din principalele deziderate ale tehnologiei **Swing** a fost să pună la dispoziție un set de componente GUI extensibile care să permită dezvoltarea rapidă de aplicații Java cu interfața grafică competitivă, din punct de vedere comercial. Cel mai important pachet, care conține componentele de bază este **javax.swing**.

Orice interfața utilizator Java este compusă din următoarele elemente:

- **Componente** – orice poate fi plasat pe o interfața utilizator, cum ar fi *butoane, liste de derulare, meniuri pop-up, casete de validare sau câmpuri de text*;
- **Containere** – acestea reprezintă componente care pot conține alte componente (de exemplu *panouri, casete de dialog sau ferestre independente*);
- **Administratori de dispunere** – reprezintă obiecte care definesc modul în care sunt aranjate (dispuse) componentele într-un container. Administratorul de dispunere nu este vizibil într-o interfața, însă sunt vizibile rezultatele "muncii" sale. Dispunerea componentelor interfeței este de mai multe feluri: dispunere secvențială, dispunere tabelară, dispunere marginală sau dispunere tabelară neproportională.

Crearea ferestrelor

Clasa **JFrame** este cea pe care o vom folosi pentru a crea ferestre. Ca orice alta clasa care reprezinta componente **Swing** ea se afla in pachetul **javax.swing**. Pentru ca este un container, vom folosi, de cele mai multe ori, aceasta clasa prin mostenire nu prin instantiere. Altfel spus, vom crea clase care sa reprezinta ferestre si pentru ca acestea sa devina ferestre de tip **JFrame** ele vor mosteni aceasta clasa.

```
public class Fereastră extends JFrame {
    private JButton button;

    public Fereastră (String text){
        super(text);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setMinimumSize(new Dimension (300, 200));
        getContentPane().setBackground (Color.blue);
        setLayout(new SpringLayout());
        button = new JButton("Apasa");
        add(button);
        show();
        pack();
    }

    public static void main(String args[]) {
        Fereastră f = new Fereastră("Laborator P00" );
    }
}
```

Metoda **add(Component c)** este folosita pentru a adauga pe fereastră o componenta exact ca in cazul Appleturilor. Metoda **add()** este mostenita din clasa **Container**.

Crearea butoanelor

Un buton poate fi creat folosind clasa **JButton**. De obicei, butonul este contruit folosind unul dintre constructorii:

- public JButton(); → un buton fara text
- public JButton(String text) → un buton cu text dat ca parametru
- public JButton(String text, Icon ico) → buton cu text si imagine

Textul de pe buton poate fi modificat folosind metoda **setText(String text)** sau poate fi preluat folosind metoda **getText()**. Metodele **setLabel()** si **getLabel()** sunt considerate obsolete si nu se mai folosesc in prezent.

```
class Button extends JFrame implements ActionListener{
    private JButton button;

    public Button(String text){
        super(text);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setMinimumSize(new Dimension(300, 200));
        getContentPane().setBackground(Color.blue);
        setLayout(new SpringLayout());
        button = new JButton("Apasa");
        button.addActionListener(this);
        add(button);
        show();
        pack();
    }

    public static void main (String args[]){
        Button b = new Button ("LaboratorP00");
    }

    @Override
    public void actionPerformed(ActionEvent e){

        /*
```

```

        JButton button = (JButton)e.getSource();
        if (button.getText().equals("Apasa")) // valabil daca aveam mai multe butoane ascultate de acest ascultator
        /*
            System.out.println("Butonul a fost apasat ! " );
        */
    }
}

```

Din moment ce butonul reprezinta o componenta, poate fi adaugat pe un container folosind metoda **add(Component c)** a containerului. Butonul **JButton** este sensibil la evenimente de tip **ActionEvent**, asadar, modalitatea de atasare a ascultatorilor si de creare evenimente este similara butoanelor **Button** din pachetul **java.awt**.

Crearea componentelor text

```

class Text extends JFrame implements ActionListener{
    private JButton button;
    private JTextField user;
    private JPasswordField pass;

    public Text (String text){
        super(text);
        setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        setMinimumSize(new Dimension(300, 200));
        getContentPane().setBackground(Color.blue);
        setLayout(new FlowLayout());
        button = new JButton ("Apasa");
        button.addActionListener(this);
        add(button);
        user = new JTextField(15);
        add(user);
        pass = new JPasswordField(15);
        add(pass);
        show();
        pack();
    }

    public static void main (String args[]){
        Text b = new Text ("LaboratorP00");
    }

    @Override
    public void actionPerformed(ActionEvent e){
        /*
            JButton button = (JButton)e.getSource();
            if (button.getText().equals("Apasa")) // valabil daca aveam mai multe butoane ascultate de acest ascultator
            */
        System.out.println(user.getText() + pass.getText());
    }
}

```

Vom folosi trei tipuri de componente text: **JTextField**, **JTextArea** si **JPasswordField**. Componenta **JTextFiled** este un camp de text cu un singur rand pe care pot incapa mai multe caractere. In general, aceasta componenta este folosita pentru a introduce un text de dimensiuni mici. Componenta **JTextArea** este un camp de text care permite introducerea unui text pe mai multe randuri, deci, aceasta componenta este folosita pentru texte de dimensiuni mai mari. Componenta **JPasswordField** este similara componentei **JTextField**, doar ca, aceasta ascunde caracterele introduse de utilizator ca in cazul unui camp pentru introducerea parolei obisnuit.

Metoda folosita pentru preluarea textului dintr-o componenta text este: **getText()** care returneaza o instanta de **String**. Pentru a modifica textul dintrun camp de text, se foloseste metoda **setText(String text)**. Componenta **JTextArea** are in comportament si metoda **append(String text)** care permite adaugarea unui text la componenta.

Butoane de selectie

C clasele **JCheckBox** si **JRadioButton** definesc componente de tip butoane de selectie. Standard, butoanele **JCheckBox** sunt folosite pentru a crea liste de optiuni de tip multiple-choice (din care poti selecta mai

multe variante), iar **JRadioButton** pentru crearea de liste de tip single-choice (din care se poate selecta o singura optiune). Astfel, pentru butoanele **JRadioButton** adaugam o noua clasa **ButtonGroup** cu ajutorul careia precizam care sunt butoanele din care se selecteaza o singura optiune. **ButtonGroup** defineste un grup de butoane. Din butoanele ce apartin aceluasi grup, nu putem selecta decat o singura optiune.

Ambele tipuri de componente se creeaza similar butoanelor **JButton**. In general, acestea nu sunt folosite cu evenimente si prin verificari ale starilor acestora (se verifica daca este sau nu select la un anumit eveniment). Verificarea starii unei astfel de componente se face folosind metoda **isSelected()** care returneaza **true** daca butonul este bifat si **false** daca nu este bifat.

Pentru o intelegere mai buna, se recomanda analizarea exemplelor propuse in arhiva laboratorului.

JScrollPane

JScrollPane este o componenta folosita pentru adaugarea de bare de defilare pentru o alta componenta care ar putea depasi dimensiunile containerului in care este adaugata. Instanta **JScrollPane** nu este folosita in actiunile directe pe care le are componenta pe care o sustine. La construire instantei, i se da o componenta pentru care este creata si pe care adauga bare de defilare (scrollbars). Apoi, instantia **JScrollPane** este adaugata pe container in locul componentei din instantia **JScrollPane**.

```
class Text extends JFrame{
    private JTextArea textArea;
    private JScrollPane scroll;

    public Text (String text){
        super(text);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setMinimumSize(new Dimension(300, 200));
        getContentPane().setBackground(Color.blue);
        setLayout(new FlowLayout());
        textArea = new JTextArea(200, 100);
        textArea.setLineWrap(true);
        textArea.setWrapStyleWord(true);
        textArea.setFont(new Font("Tahoma", 2, 12));
        scroll = new JScrollPane(textArea);
        add(scroll);
        show();
        pack();
    }

    public static void main (String args[]) {
        Text b = new Text("LaboratorP00");
    }
}
```

Etichete

O eticheta reprezinta cea mai simpla componenta. Aceasta este folosita pentru afisarea unui text static in cele mai multe cazuri. Textul de pe o eticheta poate fi modificat sau preluat folosind metodele **getText()** si **setText(String text)**.

Un **JLabel** poate fi uzitat si pentru afisarea unei imagini. In acest caz, nu i se aplica niciun text, ci doar un **imageIcon**, aceasta modalitate fiind considerata cea mai simpla de a afisa o imagine intr-un container.

Tratarea evenimentelor

Un eveniment este produs de o actiune a utilizatorului asupra unei componente grafice si reprezinta mecanismul prin care utilizatorul comunica efectiv cu programul. Exemple de evenimente sunt: apasarea unui buton, modificarea textului intr-un control de editare, inchiderea sau redimensionarea unei ferestre, etc. Componentele care genereaza anumite evenimente se mai numesc si surse de evenimente.

Interceptarea evenimentelor generate de componentele unui program se realizeaza prin intermediul unor clase de tip listener (ascultator, consumator de evenimente). In Java, orice obiect poate "consuma"

evenimentele generate de o anumita componenta grafica.

Avand in vedere gama larga de evenimente existente in limbajul Java, am ales sa vi le prezint pe cele considerate mai important, intr-o serie de exemple incluse in arhiva laboratorului.

poo/breviare/breviar-10.txt · Last modified: 2020/12/10 13:47 by carmen.odubasteanu